Computing with functions on domains with arbitrary shapes

Daan Huybrechs
Roel Matthysen

Report TW 678, June 9, 2017

KU Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Computing with functions on domains with arbitrary shapes

Daan Huybrechs
Roel Matthysen

Report TW 678, June 9, 2017

Department of Computer Science, KU Leuven

Abstract

We describe an approximation scheme and an implementation technique that enables numerical computations with functions defined on domains with an arbitrary shape. The scheme is spectrally accurate for smooth functions. The main advantage of the technique is that, unlike most spectral approximation schemes in higher dimensions, it is not limited to domains with tensor-product structure. The scheme itself is a discrete least squares approximation in a redundant set (a frame), that originates from a basis on a bounding box. The implementation technique consists of representing a domain by its characteristic function, i.e. the function that indicates whether or not a point belongs to the set. We show in a separate paper that the least squares approximation with N degrees of freedom can be solved in $\mathcal{O}(N^2 \log^2 N)$ operations for any domain that has non-trivial volume. The computational cost improves to $\mathcal{O}(N \log^2 N)$ operations for domains that do have tensor-product structure. The scheme applies to domains even with fractal shapes, such as the Mandelbrot set, since such domains are defined precisely by their characteristic function

Keywords : Fourier Series, Fourier Extensions, Fourier Continuation, Prolate Spheroidal Wave Theory.

Computing with functions on domains with arbitrary shapes

Daan Huybrechs and Roel Matthysen

Abstract We describe an approximation scheme and an implementation technique that enables numerical computations with functions defined on domains with an arbitrary shape. The scheme is spectrally accurate for smooth functions. The main advantage of the technique is that, unlike most spectral approximation schemes in higher dimensions, it is not limited to domains with tensor-product structure. The scheme itself is a discrete least squares approximation in a redundant set (a frame), that originates from a basis on a bounding box. The implementation technique consists of representing a domain by its characteristic function, i.e. the function that indicates whether or not a point belongs to the set. We show in a separate paper that the least squares approximation with N degrees of freedom can be solved in $\mathcal{O}(N^2 \log^2 N)$ operations for any domain that has non-trivial volume. The computational cost improves to $\mathcal{O}(N \log^2 N)$ operations for domains that do have tensor-product structure. The scheme applies to domains even with fractal shapes, such as the Mandelbrot set, since such domains are defined precisely by their characteristic function.

1 Introduction

1.1 Motivation

Approximation schemes of a single variable usually extend to multivariate schemes only when the domains under consideration have tensor-product structure, the simplest domains being squares and rectangles. Functions on more general domains

Daan Huybrechs
KU Leuven, e-mail: daan.huybrechs@kuleuven.be

Roel Matthysen
KU Leuven e-mail: roel.matthysen@kuleuven.be

can be approximated using mappings of tensor-product domains to more general domains, or by tiling a general domain with quadrilaterals. These approaches are powerful and they are fundamental in scientific computing, though usually only low-order approximations are aimed for in that setting. Moreover, the complexity of the mapping or of the tiling clearly depends on the complexity of the geometry of the domain under consideration. This affects the performance of the scheme, even when the function to be approximated is perfectly smooth and, from an approximation point of view, 'well-behaved'.

2 The approximation scheme

Let the domain of interest be a compact subset B of $\mathbb{R}^n$. Let f be a target function of n variables in $\mathscr{L}^2_B$, the space of functions that are square integrable over B. We represent f through a basis $G = \{\phi_i\}$ for $\mathscr{L}^2_C$, where C is a domain that contains B in its interior and for which a basis can easily be found. Further on, we will use a bounding box and define a classical tensor-product basis on that box. In general, when a basis for $\mathscr{L}^2_C$ on C is restricted to a subdomain B, the resulting set is a frame for $\mathscr{L}^2_B$ in the sense of Duffin and Schaeffer [5].

Assuming some linear ordering of the elements of G, we denote a truncated frame by $G_N = \{\phi_i\}_{i=1}^N$ and its span by $\mathscr{G}_N = \operatorname{span} G_N$. We define the best approximation to f in this space using the associated norm over B,

$$f_N = \arg\min_{g\in\mathscr{G}_N} \|f-g\|_B. \tag{1}$$

This results in increasingly accurate approximations in the truncated set G_N, as N increases.

In order to arrive at a simple linear system, we substitute the $\mathscr{L}_2$ norm over B by a discrete norm over a point set P_M, consisting of $M \geq N$ points in B,

$$f_N = \arg\min_{g\in\mathscr{G}_N} \sum_{\mathbf{x}\in P_M} |f(\mathbf{x})-g(\mathbf{x})|^2. \tag{2}$$

Here we assume that besides being in $\mathscr{L}^2_B$, f is also continuous on B. With a similar linear ordering of the point set P_M this leads to a least squares system

$$A\alpha = b, \qquad A_{ij} = \phi_j(\mathbf{x}_i), \qquad b_i = f(\mathbf{x}_i). \tag{3}$$

The matrix A has dimensions $M \times N$, and further on we typically choose $M = 2N$. We stress that both the norm in (1) and the points in (2) are confined to B. This implies that we make no assumption about the existence of f outside of the domain B, i.e., we require no information from f on the extension region $C \setminus B$.

There are very few practical restrictions on G_N, besides completeness in $\mathscr{L}^2_B$. A disadvantage compared to using a basis tailored to the domain B is that the condi-

tioning of A in eq. (3) can be arbitrarily bad. In fact, the expansion $f_N = \sum_{i=1}^{N} \alpha_i \phi_i$ might not even be unique and A can be singular.

Both the ill-conditioning and the potential lack of uniqueness correspond to the typical redundancy of a frame compared to a basis. From an approximation point of view, both of these effects are relatively benign. The numerical stability of this type of least squares approximation was studied in [2] for the case where B is an interval and G is a Fourier basis on a larger interval. This analysis is generalized to numerical approximations in more general frames in [1]. The discrete least squares approximation with oversampling ($M > N$) leads, somewhat surprisingly, to a stable approximation scheme for sufficiently large N, regardless of the shape of the domain $B \subset C$.

A compelling practical advantage of the present approach is that constructing bases for arbitrary domains is hard in general, and requires at least some a priori domain knowledge. In contrast, our scheme can be based on any known classical basis that spans $\mathscr{L}^2$ on a bounding box C encompassing B. It requires only the generation of a suitable point set P_M, which we discuss further on.

Because of ill-conditioning of A, iterative solvers tend not to perform very well in these applications. Direct solvers, and in particular a truncated singular value decomposition which allows for some regularization of the solution, seem approriate. However, they come at an $\mathscr{O}(N^3)$ cost.

2.1 The Fourier extension scheme

For certain choices of bases and point sets the approximation scheme allows for very efficient solutions to eq. (3). When the approximation space is that of periodic functions on C, and the point set is the intersection of an equispaced n-dimensional grid with B, the scheme is known as *Fourier Extension*. An illustration showing the equispaced grid restricted to a domain is shown in fig. 1.

The least squares matrix A in this case can be separated into a well-conditioned part, and a lower-rank part that captures the ill-conditioning [8, 9, 10]. By solving the lower-rank part first, and the well-conditioned part quickly, the total complexity becomes $\mathscr{O}(N^{3-2/n}\log^2(N))$. This is linear up to a logarithmic term for $n = 1$, and can provide a substantial speedup in higher dimensions for sufficiently large N.

The fast algorithms of [9, 10] can also be employed when using Chebyshev polynomials on the bounding box and an associated tensor product grid of Chebyshev points, restricted to the domain B.

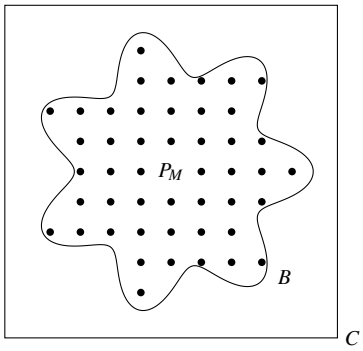


Fig. 1: The bounding box C encompassing the equidistant samples P_M.

2.2 Computing with a spectral basis

As demonstrated by the Chebfun package [4], expansions in a spectral basis such as those resulting from the Fourier Extension allow you to compute with functions by manipulating the expansion coefficients. For example:

- Scalar multiplication and summation of approximations is performed in coefficient space.
- Pointwise multiplication of two Fourier series can be accomplished through convolution of coefficients.
- Constant coefficient differential operators are easy to apply in coefficient space. In the Fourier case, these operators are diagonal.
- Rootfinding and finding extrema can be found by applying existing methods for the bounding box, and restricting the results to the domain of interest.

Limitations are that due to the nature of the Fourier basis singularities are difficult to deal with, and restrict convergence rates. This may be ameliorated by adding suitable singularity functions to the frame, resulting in a slightly larger frame with better approximation properties [1]. Another limitation is that the expansion on the bounding box is naturally defined on the bounding box and not just on the domain. It is simple to compute the definite integral over the bounding box, but not straightforward to do so over the embedded domain.

The third bullet above (diagonalization of constant coefficient differential operators) make Fourier extensions an appealing scheme for boundary value problems on complicated domains. The use of implicit or explicit extensions to treat complex geometries arises already in several schemes in scientific computing, such as embedded domain methods, immersed or fictitious boundary methods and volume-penalty methods [3, 7, 11]. The difference compared to our setting is that with oversampled Fourier extensions we obtain spectral accuracy with an efficient algorithm. Our current efforts in this direction are related to meshless methods, in particular to Kansa's method often used with radial basis functions [6].

3 Domain representation

3.1 The characteristic function

The *characteristic function* χ, or *indicator function*, of a domain $B \subset \mathbb{R}^n$ is a function on $\mathbb{R}^n$ that has value 1 for points that belong to B and the value 0 for points that do not, i.e.,

$$\chi(x) := \begin{cases} 1, & x \in B, \\ 0, & x \notin B. \end{cases} \tag{4}$$

It is convenient in implementations to associate boolean values with $\chi(x)$, so that it evaluates to true or false, rather than the numeric values 1 and 0.

Representing a domain by its characteristic function has a number of consequences. Two **advantages** are:

- The function is unique and well-defined for any domain, be it open or closed, connected or disconnected, punctured, empty, a discrete set, finite or infinite, a fractal, ...
- As we will see later on, the characteristic function is often easy to implement. For example, with $x = [x_1, x_2]$ in two dimensions, the halfopen domain bounded by the parabola $x_2 = x_1^2$ and the straight line $x_2 = x_1$ has characteristic function

$$\chi(x) = (x_2 > x_1^2) \,\&\, (x_2 \leq x_1). \tag{5}$$

  There is no need even to find the intersection points of both curves, as far as implementing the characteristic function is concerned.

Consequently, it is easy and very cheap to find the characteristic function of the domain that is bounded by, say, the level curves of a given function, even if the resulting domain is disconnected and contains many holes. This operation does not even require any numerical computation, as will be demonstrated later on.

Two **disadvantages** are:

- The characteristic function does not explicitly convey information about the boundary of the domain. This would be difficult for fractal domains, but it would be convenient to have for simpler domains, and essential to have for boundary value problems.
- The least squares approximation scheme requires point evaluations inside the domain. Though the characteristic function is well-defined for domains that have no volume in $\mathbb{R}^n$, such as a line in $\mathbb{R}^2$ or a surface in $\mathbb{R}^3$, the concept is not suited for approximating functions on such domains.

3.2 Generating points

The least squares approximation scheme requires M point evaluations of the given function f inside the domain B. Thus, one needs a way to find M points that belong to B.

It is convenient at this stage too to have at hand a bounding box C, or the knowledge of any other region C that is easily sampled for which $B \subset C$. Then, points inside B can be generated by sampling Q points y_j of C and checking whether $\chi_B(y_j)$ is true. This results in a set of M_Q points with $M_Q \leq Q$:

$$\{x_j\}_{j=1}^{M_Q} := \{y_j \,|\, \chi_B(y_j) = 1, j = 1, \ldots, Q\}$$

Only those points are retained and the procedure is repeated with denser samplings, corresponding to increasing values of Q, until $M_Q \geq M$ points are retained.

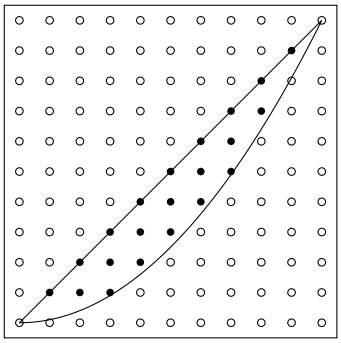


Fig. 2: The characteristic function (5) evaluated in Q points inside the bounding box C. The points are a subset of a structured equispaced grid on C.

For domains with non-zero volume in $\mathbb{R}^n$, it is guaranteed that M points will eventually be found if the sampling of C becomes uniformly denser. Though in principle any randomly chosen set of points $\{x_j\}$ is sufficient, for efficiency reasons it is better to choose a structured set. In particular, in combination with the Fourier extension scheme we use a bounding box C and an equidistant grid on C. The main advantage is that Fourier series approximations can be evaluated efficiently on that grid with the FFT. In several examples further on, the characteristic function of a domain itself is defined in terms of a Fourier series, and in that case too the characteristic function too can be evaluated efficiently on a structured grid using the FFT.

From the point of view of the approximation problem, it may be better to have more points clustered towards the boundary of the domain. However, even for multivariate polynomial interpolation it is a very difficult problem to determine the best points on a general domain. Furthermore, since we make very few assumptions re-

garding our knowledge of the boundary, choosing more points near the boundary requires algorithmic work. Instead, we oversample.

3.3 Implementation

In an object-oriented approach, a domain would be an object. Disregarding that property, the elements that make up a domain include:

- a bounding box,
- a routine to evaluate the characteristic function at a single point x,
- an optimized routine to evaluate the characteristic function on a regular grid of the bounding box.

The latter routine will be called the *grid evaluation* routine. It is not an essential part of the implementation, but it leads to much increased efficiency in particular when using the Fourier extension approximation scheme. The goal is not merely to exploit the speed gained from vectorization, but to lower the computational complexity compared to calling the single evaluation routines many times.

For points on the boundary, the characteristic function can be true or false, corresponding to closed and open domains. This makes a difference in practice only in special circumstances, since in general the points that are sampled are unlikely to coincide with the boundary of the domain. In general, it is very difficult to distinguish between open and closed domains with the proposed techniques.

We have implemented the scheme in Julia[1], with a user interface that is modelled after the software package Chebfun [4]. In our implementation the domain is implemented as an object with three properties, corresponding to the three elements above.

4 Computing with domains

4.1 Set operations

Basic set operations have rather obvious ramifications for the characteristic function. The union, difference and intersection of two domains give rise to logical relationships between the characteristic functions involved. Assume the domains B_i, $i = 1,2,3$, have characteristic functions χ_i. Then we have

[1] The code is publicly available in the online GitHub repository of the FrameFuns package, `http://github.com/daanhb/FrameFun.jl`.

$$B_3 = B_1 \cup B_2 \Rightarrow \chi_3(x) = \chi_1(x) \textbf{ or } \chi_2(x)$$
$$B_3 = B_1 \cap B_2 \Rightarrow \chi_3(x) = \chi_1(x) \textbf{ and } \chi_2(x)$$
$$B_3 = B_1 \setminus B_2 \Rightarrow \chi_3(x) = \chi_1(x) \textbf{ and not } \chi_2(x)$$
$$B_3 = (B_1 \cup B_2) \setminus (B_1 \cap B_2) \Rightarrow \chi_3(x) = \chi_1(x) \textbf{ xor } \chi_2(x).$$

These operations are easily implemented by defininining χ_3 in terms of the supplied definitions of χ_1 and χ_2. Similarly, the grid evaluation routine of B_3 can be defined in terms of the grid evaluation routines in B_1 and B_2. This makes sure that a potentially fast implementation of this procedure for B_1 and B_2 leads to a fast implementation of this procedure also for B_3.

In Julia, this enables the following operations

```
>> B3 = B1 & B2
>> B3 = B1 | B2
>> B3 = B1 \ B2
>> B3 = xor(B1,B2)
```

by overloading the logical operators for domain objects.

4.2 Arithmetic operations

Domains can be translated and scaled by adding a vector and by multiplying by a scalar respectively. We have

$$\forall c \in \mathbb{R}^n : \quad B_2 = B_1 + c \Rightarrow \chi_2(x) = \chi_1(x-c)$$
$$\forall a \in \mathbb{R} : \quad B_2 = a * B_1 \Rightarrow \chi_2(x) = \chi_1(x/a).$$

It should be noted that while translation of a domain is independent of the location of the origin, scaling a domain like above does depend on the location of the origin. A circle centered around the origin would simply increase in size by a factor of a, but a circle centered at the point $[1;0]$ would also move a factor a to the right.

Arithmetic operations are also easily implemented, by defininining χ_2 in terms of the supplied definition of χ_1 and similarly for the grid evaluation routines.

In Julia we may write

```
>> B2 = B1 + [1;0]
>> B3 = 2*B1
```

Combined with the above, a moon-shaped domain can be defined in terms of a circle C with radius 1 by the statement

```
>> moon = C \ (C + [1/2; 0])
```

Similarly, if C is centered around the origin, a domain with a hole is obtained by

```
>> annulus = 2*C \ C
```

4.3 Implicitly defined or derived domains

Finding the level curves of a function, say the set of points where $f(x) = 3$, requires algorithmic work and can become arbitrarily complicated depending on the complexity of the given function f. However, it is very easy to define the characteristic function of a domain that is bounded by this level curve. Say a function f is defined on B and the domain C is the open domain where $f(x) > 3$. Then the characteristic function χ_C of C is given explicitly by

$$\chi_C(x) = \begin{cases} f(x) > 3, & \forall x \in B, \\ 0, & \text{otherwise.} \end{cases}$$

The implementation of the characteristic function is defined in terms of the inequality $f(x) > 3$, which is a boolean expression for each x. The grid evaluation routine of C may be implemented in terms of the grid evaluation routine of f. Thus, if f can be evaluated efficiently via FFT for example, then the same holds for the characteristic function of the domain C.

In Julia, we may now write

```
>> C = f > 3
>> C = f > g
>> C = cos(f .^ 2) - 3 < sqrt(pi)
```

where both f and g are existing functions. In the second statement, the domain C is in addition restricted to the intersection of the domains of f and g, such that it makes sense to compare f and g.

Interestingly, from the point of view of implementation, it is irrelevant whether or not the resulting domains are connected or not. The shape of the resulting domain can be truly arbitrary and does not effect the computational cost of this new characteristic function. Of course, the geometry of the domain does play a role in the approximation problem to be solved, though even there its influence remains fairly minor.

4.4 Deciding on the equivalence of domains

When given two characteristic functions χ_1 and χ_2, the problem of deciding whether they represent the same domain is a difficult one and requires careful consideration.

It is of course not possible to check for each and every point $x \in \mathbb{R}^2$ whether $\chi_1(x)$ equals $\chi_2(x)$. Two possible ways to treat this problem are as follows.

1) Verify equivalence up to a certain resolution

The characteristic functions χ_1 and χ_2 are sampled on an equidistant grid with a certain specified resolution and covering both domains. Their equivalence at this resolution level is determined by their equivalence at the grid points.

2) Construct a global table of domains with unique identifiers

The domain object can be extended with a unique identifier. Domains are compared by comparing their identifiers. Each operation that results in a new domain also results in a new identifier, which is kept in a global table. A function g that is computed from a function f inherits the domain of f, along with its identifier.

The first approach is costly and does not always give the right mathematical answer, in the sense that it may conclude equivalence for two domains that are not equivalent. It will never conclude inequivalence for equivalent domains. However, the approach applies to all domains and will always converge to the correct answer when increasing the resolution level.

The second approach is fast, but comes at a cost of having to construct a global table. This adds overhead and memory costs. The approach also does not always give the right mathematical answer, as two domains may be constructed in similar ways but independently of each other. Their identifiers will be different, though the domains may be the same. Avoiding this situation requires care from the user of the software.

5 Examples

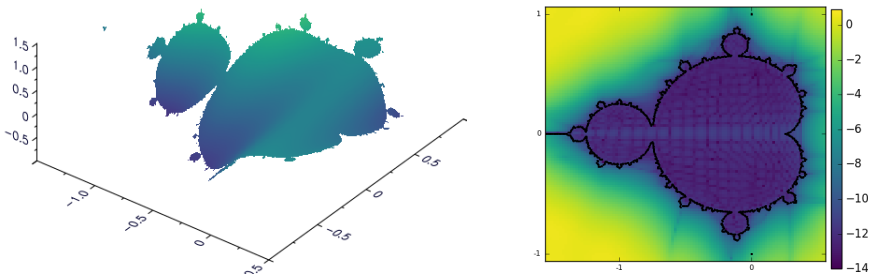
5.1 Characteristic function

For some domains the characteristic function is simply the most convenient description. The Mandelbrot set is an example, defined by

$$\chi(x) = \left(\limsup_{n\to\infty} |z_{n+1}| \leq 2 \right),$$
$$z_{n+1} = z_n^2 + x_1 + \mathrm{i}x_2, \qquad z_0 = 0.$$

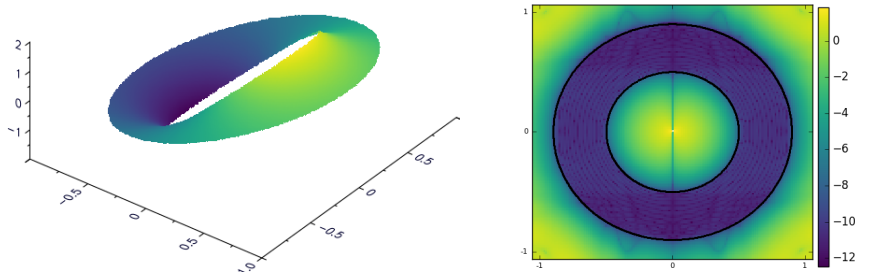
An approximation of

$$F_m(x) = \cos(20x_1 + \mathrm{i}x_2) - 5x_1x_2 \tag{6}$$



(a) Approximation (b) Error

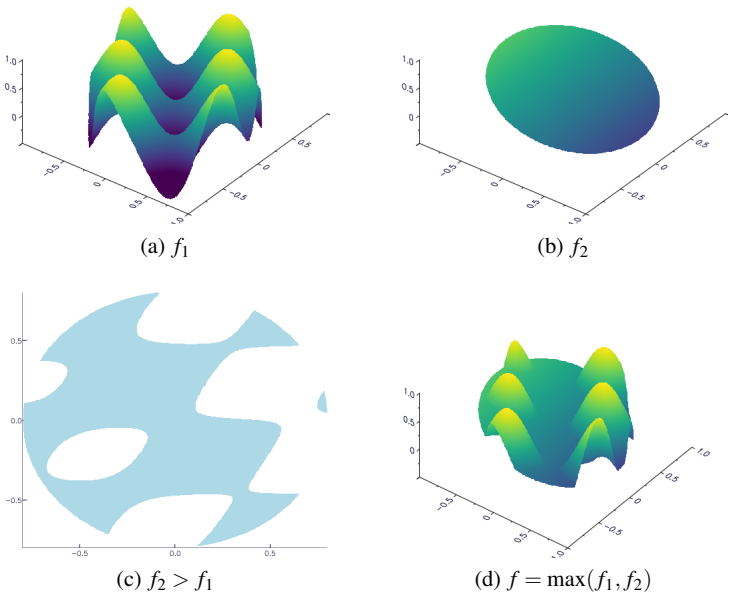
Fig. 3: An approximation of F_m (eq. (6)) on the Mandelbrot set. The right figure shows $\log_{10}(|f_N - F_m|)$. The approximation error is very small precisely on the Mandelbrot set. In the extension region, the functions f_N and F_m are both defined and they can be evaluated and compared, but they bear no resemblance. In particular, f_N is periodic on the box, while F_m is not.



(a) Approximation (b) Error

Fig. 4: An approximation of F_r (eq. (7)) on a ring-shaped domain. The right figure shows $\log_{10}(|f_N - F_r|)$.

is shown in fig. 3a. It was obtained using an equispaced grid on $[-2,2] \times [-1.5,1.5]$. Using the Fourier Extension technique, convergence up to a tolerance of 10^{-12} was achieved for 32×32 basis functions (fig. 3b).



(a) f_1 (b) f_2

(c) $f_2 > f_1$ (d) $f = \max(f_1, f_2)$

Fig. 5: A piecewise approximation of F (eq. (10)), and the implicit domain $f_2 > f_1$.

5.2 Domain arithmetic

As an example of computing with domains, fig. 4 shows an approximation on a ring, obtained by the Julia commands

```
>> B3 = Disk(0.9) \ Disk(0.5)
```

Of special note here is that the target function

$$F_r(x) = \frac{x_1}{x_1^2 + x_2^2} \tag{7}$$

has a singularity at $(0,0)$, enclosed in the domain. However, this has no influence on the approximation, as the exterior of the domain is never sampled. As fig. 4b shows the approximation converges up to a tolerance of 10^{-10} for 32×32 basis functions.

5.3 Implicitly defined domains

More convoluted domains occur when trying to approximate a function such as

$$F_1(x) = \sin(5x_1 - 3x_2)\sin(7x_1) \tag{8}$$
$$F_2(x) = -0.5x_1 + 0.2 \tag{9}$$
$$F(x) = \max(F_1(x), F_2(x)) \tag{10}$$

on a disk. Given f_1 and f_2, approximations of F_1 and F_2 on the full disk (figs. 5a to 5b), f_N is simply

$$f_N(x) = \begin{cases} f_1(x), & f_1(x) \geq f_2(x) \\ f_2(x), & f_1(x) < f_2(x) \end{cases}.$$

In this case, evaluating f_N (fig. 5d) or the characteristic function (fig. 5c) is straightforward, and fast on an equispaced grid, since only one full evaluation of f_1 and f_2 is required.

Acknowledgements

The authors are supported by FWO Flanders Projects G.A004.14 and G.0641.11

References

1. B. Adcock and D. Huybrechs. Frames and numerical approximation. Technical Report TW-674, KU Leuven, December 2016.
2. B. Adcock, D. Huybrechs, and J. Martín-Vaquero. On the numerical stability of Fourier extensions. *Found. Comp. Math.*, 14:635–687, 2014.
3. D. Boffi, N. Cavallini, and L. Gastaldi. The finite element immersed boundary method with distributed Lagrange multiplier. *SIAM J. Numer. Anal.*, 53(6):2584–2604, 2015.
4. T. A. Driscoll, N. Hale, and L. N. Trefethen. *Chebfun guide*. Pafnuty publications, Oxford, 2014.
5. R. J. Duffin and A. C. Schaeffer. A class of nonharmonic Fourier series. *Trans. Amer. Math. Soc.*, 72(2):341–366, 1952.
6. E. J. Kansa. Multiquadrics – a scattered data approximation scheme with applications to computational fluid-dynamics II: solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & Mathematics with Applications*, 19:147–161, 1990.
7. F. Kasolis, E. Wadbro, and M. Berggren. Analysis of fictitious domain approximations of hard scatterers. *SIAM J. Numer. Anal.*, 2015(5):2347–2362, 2015.
8. M. Lyon. A fast algorithm for Fourier continuation. *SIAM J. Sci. Comput.*, 33(6):3241–3260, Jan. 2011.
9. R. Matthysen and D. Huybrechs. Fast Algorithms for the computation of Fourier Extensions of arbitrary length. *SIAM J. Sci. Comput.*, 36(2):828–845, 2015.
10. R. Matthysen and D. Huybrechs. Multi-dimensional Fourier Frame Approximations through Collocation. *In preparation*, 2016.
11. D. Shirokoff and J.-C. Nave. A sharp-interface active penalty method for the incompressible Navier–Stokes equations. *J. Sci. Comput.*, 62(1):53–77, 2015.